

User Guide



December 2013

National Institute of Standards and Technology (NIST)
Information Technology Laboratory (ITL)
Information Access Division (IAD) / Image Group

Disclaimer for Video-based Automated System for Iris Recognition (VASIR)

December 17, 2013

The software was developed by the National Institute of Standards and Technology (NIST), an agency of the Federal Government. Pursuant to Title 15 United States Code Section 105, works of NIST are not subject to copyright protection in the United States and are considered to be in the public domain. Thus, the software may be freely reproduced and used. Please explicitly acknowledge the National Institute of Standards and Technology as the source of the software.

This software is released by NIST as a service and is expressly provided "AS IS." NIST MAKES NO WARRANTY OF ANY KIND, EXPRESS, IMPLIED OR STATUTORY, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTY OF MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE, NONINFRINGEMENT AND DATA ACCURACY. NIST DOES NOT REPRESENT OR WARRANT THAT THE OPERATION OF THE SOFTWARE WILL BE UNINTERRUPTED OR ERROR-FREE, OR THAT ANY DEFECTS WILL BE CORRECTED.

Certain trade names and company products are mentioned in the text or identified. In no case does such identification imply recommendation or endorsement by the National Institute of Standards and Technology, nor does it imply that the products are necessarily the best available for the purpose.

With the exception of material marked as copyrighted, information presented in this document is considered public information and may be distributed or copied. Use of appropriate by line/photo/image credits is requested.

Contents

Disclaimer for Video-based Automated System for Iris Recognition (VASIR)	2
Purpose	5
Requirements.....	6
Configuration	6
Prerequisites	6
1. Qt Toolkit	6
2. Qt Creator	6
3. OpenCV Library (Version 2.3.1)	6
For Windows	6
For Mac	6
4. VASIR source code (Written in C++)	7
Installation	7
1. Qt Installation	7
For Windows with Visual Studio	7
For Mac	7
2. Qt Creator for both Windows and Mac	7
3. OpenCV Installation	8
For Windows with Visual Studio	8
For Mac	8
4. Running VASIR source code for both Windows and Mac.....	9
Getting Started.....	10
Cascade menu	10
Detect menu	10
Extracting the eye region image	12
Match tab.....	13
Quality menu.....	16
Match menu.....	16
Analysis menu	17
Annex A. Performance on Datasets	19
Datasets	19
Performance Evaluation	19
Annex B. Doxygen description of source code	19

Purpose

This document gives an overview on how to operate VASIR. The discussed version contains several analysis functions. A link to the documented source-code and information regarding the datasets used can be found in the Appendix.

The tool can be used to:

- Detect the eye pair (left and right eyes) within their face-visible video data
- Provide the left and right preocular or iris region for image quality analysis
- Identify a person using the iris feature
- Advance still- and video-based (or real-time) iris recognition
- Study the sensitivity and robustness analysis
- Conduct an optimization study

We use the “Qt Creator” software that allows us to manage the VASIR system on both Windows and Unix operating systems without any changes of the VASIR code base.

The system is under on-going development and optimization efforts. Therefore, it is to be understood that the source code produces warnings and is not without bugs. We plan to update each component of VASIR over time after evaluating its quality and performance.

Please note that we cannot warrant the correctness, usefulness, accuracy, reliability, etc. of the source code. We would, however, appreciate if you could send ANY COMMENTS and BUG REPORTS to vasir@<NOSPAM>nist.gov.

Please cite the paper below as you use the VASIR system.

Yooyoung Lee, Ross J. Micheals, P. J. Phillips, James J. Filliben, “VASIR: An Open-Source Research Platform for Advanced Iris Recognition Technologies”, Journal of Research, National Institute of Standards and Technology (JRNIST), V118, p218-259, 2013. <http://dx.doi.org/10.6028/jres.118.011>

Requirements

Configuration

The tool has been tested on the following Operating Systems:

- Microsoft Windows XP Professional Edition
- Microsoft Windows Server 2008
- Microsoft Windows 7 (x86)
- Mac OS X 10.8.4

The recommended minimum configuration is as follows:

- CPU: Dual Core, 2GHz
- RAM: 3GB
- Display resolution: 1280 x 1024 pixels

Prerequisites

1. Qt Toolkit

- Download the version 4.8.2 of the Qt Toolkit at <http://download.qt-project.org/archive/qt/4.8/4.8.2/>
- For example,
 - “Windows with Visual Studio compiler” download “[qt-win-opensource-4.8.2-vs2010.exe](#)”
 - “Windows with Mingw compiler” download “[qt-win-opensource-4.8.2-Mingw.exe](#)”
 - “Mac” download “[qt-mac-opensource-4.8.2.dmg](#)”

2. Qt Creator

- Qt Creator is shipped with Qt if you would like to download a specific version of Qt Creator, then you can download Qt Creator from: <http://qt-project.org/downloads#qt-creator>

3. OpenCV Library (Version 2.3.1)

For Windows

- Download OpenCV 2.3.1 at <http://sourceforge.net/projects/opencvlibrary/files/opencv-win/2.3.1/>
(e.g., OpenCV-2.3.1-win-superpack.exe [131MB])
- Download CMake 2.x.x from <http://www.cmake.org/cmake/resources/software.html>
(e.g., cmake-2.8.8-win32-x86.exe (9MB))

For Mac

- Please refer to the section “OpenCV Installation for Mac”.

4. VASIR source code (Written in C++)

- Download the source code “NIST_VASIR_src_beta_2.2 (ZIP, 240KB)” from <http://www.nist.gov/itl/iad/ig/vasir.cfm>

** NOTE: You may choose to use a different operating environment (e.g. Linux). However, the installation steps might vary depending on your configuration.*

Installation

1. Qt Installation

For Windows with Visual Studio

The following steps exemplify the installation of Qt 4.8.2 version on Windows 7:

- 1) Create the Qt directory (e.g., “\Qt-4.8.2\”)
- 2) Install “qt-win-opensource-src-4.8.2-vs2010” into the directory “\Qt-4.8.2\”
- 3) Open “Visual Studio Command Prompt (2010)”
- 4) Run the following commands (type w/o the leading “\$”):

```
cd \Qt-4.8.2\  
set PATH += \ Qt-4.8.2\  
set PATH += \ Qt-4.8.2\bin\  
set LIB += C:\Program Files(x86)\Microsoft SDKs\Windows\v7.0A\Lib  
set INCLUDE += C:\Program Files(x86)\Microsoft SDKs\Windows\v7.0A\Include
```
- 5) Configure FULL VERSION:
configure -platform win32-msvc2010
* Note that if you have a message “perl not found in environment - cannot run synqt”, then delete or rename “synqt.bat” under “\bin” folder and run it again.
- 6) Run “nmake” (*Note that this may take for a while)
- 7) Add “.\Qt-4.8.2\bin\” to the “PATH” in Environment variables:
- 8) System > Advanced system settings> Advanced >Environment variables>User variables

For Mac

The following steps exemplify the installation of Qt 4.8.2 version on Mac OS X 10.8.4:

- 1) Mount the disk image (.dmg)
- 2) Open the mounted disk image in Finder
- 3) Install the meta-package “Qt.mpkg”

2. Qt Creator - both Windows and Mac

- 1) Start Qt Creator
- 2) Check if Qt Creator is using the Qt version that you’ve installed
- 3) For Windows: Tools -> Options -> Builds & Runs
For Mac: Qt Creator -> Preferences -> Builds & Runs

- 4) "Compilers" tab: use either "Microsoft Visual C++ Compiler" or "MinGW" (e.g., download and install Windows SDK at <http://www.microsoft.com/en-us/download/details.aspx?id=8279>)
- 5) "Qt Versions" tab: most of the time, Qt Creator will automatically detect the Qt version that you have installed. If not auto-detected or if you would like to use a specific version instead, click the "add" button and browse the "qmake.exe" binary within the correct Qt version path.
- 6) Click "Apply" button.
- 7) "Kits" tab: select the same Qt version under "Compiler" and "Qt Version" items. For example, Kits: Add -> Qt version -> choose (e.g., Qt 4.8.2)
Debugger: click the "Edit" button -> load "CDB.exe" (e.g., c:\programe files\windows kits\8.0\Debuggers\x86\cdb.exe)
* Note that on Mac Qt Creator will automatically detect the correct debugger instance for you. In case you don't have a pre-installed debugger on windows, download and install the "Debugging tools for Windows", and then select "cdb.exe" file (e.g., download Windows SDK for Windows at <http://msdn.microsoft.com/en-US/windows/hardware/hh852363>)

3. OpenCV Installation

For Windows with Visual Studio

The following steps exemplify the installation of OpenCV 2.3.1 on Windows 7 with Visual Studio 2010:

- 1) Double click OpenCV-2.3.1-win-superpack.exe and extract all files (e.g., we changed the opencv directory to "\opencv-2.3.1")
- 2) Create a folder (e.g., OpenCV-2.3.1/opencv_binaries/)
- 3) Install cmake-2.8.8-win32-x86.exe and open
- 4) Do the following to configure:
 - Where is the source code: /opencv-2.3.1
 - Where to build the binaries': /opencv_2.3.1/opencv_binaries
 - Check box WITH_QT
 - Click "configure" button, select "Visual Studio 10", and then click "finish" button
 - Click "generate" button
- 5) Open "ALL_BUILD" under the opencv_binaries folder
- 6) Build opencv
- 7) Add "\opencv_binaries\bin\Debug" to the "PATH" in Environment variables:
- 8) System > Advanced system settings> Advanced >Environment variables>User variables

For Mac

The following steps exemplify the installation of OpenCV 2.3.1 on Mac OS X 10.8.4:

- 1) Install Homebrew (<http://brew.sh>) by running in a Terminal/shell (w/o the leading "\$"):


```
$ ruby -e "$(curl -fsSL https://raw.githubusercontent.com/mxcl/homebrew/go)"
```
- 2) Edit your "~/.profile" and add:


```
export PYTHONPATH="/usr/local/lib/python2.7/site-packages:$PYTHONPATH"
```
- 3) Add the science "tap":


```
$ brew tap homebrew/science
```

- 4) Refresh Homebrew's package information:
\$ brew update
- 5) Go back to OpenCV version 2.3.1:
\$ cd /usr/local/Library/Taps/homebrew-science/opencv.rb
\$ git checkout cdaf83d opencv.rb
- 6) Install OpenCV 2.3.1:
\$ brew install --32-bit --with-libtiff opencv

4. Running VASIR source code for both Windows and Mac

- 1) Unzip "NIST_VASIR_src_beta_v2.2.zip"
- 2) Using Qt Creator, compile the project in the following order:
 - MasekAlg
 - Step1: open the existing project file "MasekAlg.pro" under the "MasekAlg" folder
 - Step2: configure the paths for "debug" and "release" mode to "Build/MasekAlg" (e.g., click "Projects" -> Edit build configuration -> build directory -> browse "../Build/MasekAlg")
If you have a link problem, please double-check the path for LIBS and INCLUDEPATH in the "MasekAlg.pro" file
 - Step3: build the source code (e.g., "Build" menu -> click "Build All")
 - VASIR
 - Repeat steps 1-3 for VASIR
 - Analysis
 - Repeat steps 1-3 for Analysis
 - YooIRIS
 - Repeat steps 1-3 for YooIRIS
- 3) Run YooIRIS.exe (e.g., ..\Beta2.2\Build\YooIRIS\debug\YooIRIS.exe)

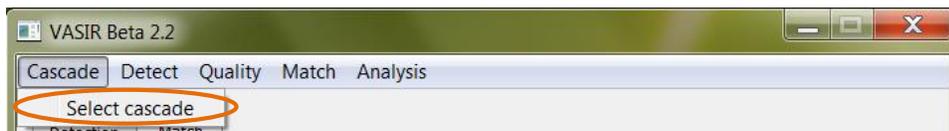
**Warning:* The source code is to be considered beta—not final. Therefore known warning or memory related messages may be shown during compilation and execution.

Getting Started

Cascade menu

The “Cascade” menu allows the user to load a Haar-classifier (XML format). This needs to be done before loading a video file. (If you want to load a still image directly, please see Section “Match tab”). “parojosG.xml” is the default classifier file. A classifier named “parojosG.xml” can be found in “bin/cascade/” within the default folder.

Click “Select cascade” and load the required classifier.



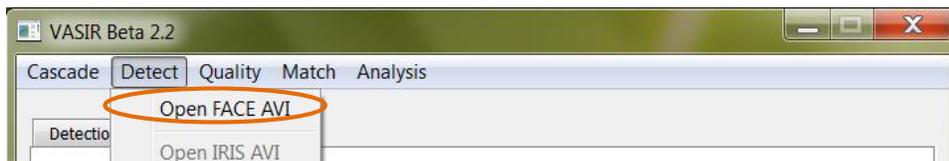
Note: Depending on your preference, a different or custom classifier may be used.

Detect menu

The “Detect” menu allows the user to load a video file in AVI format. Both the “Open IRIS AVI” and “Open CAM” entry are currently disabled for future use.

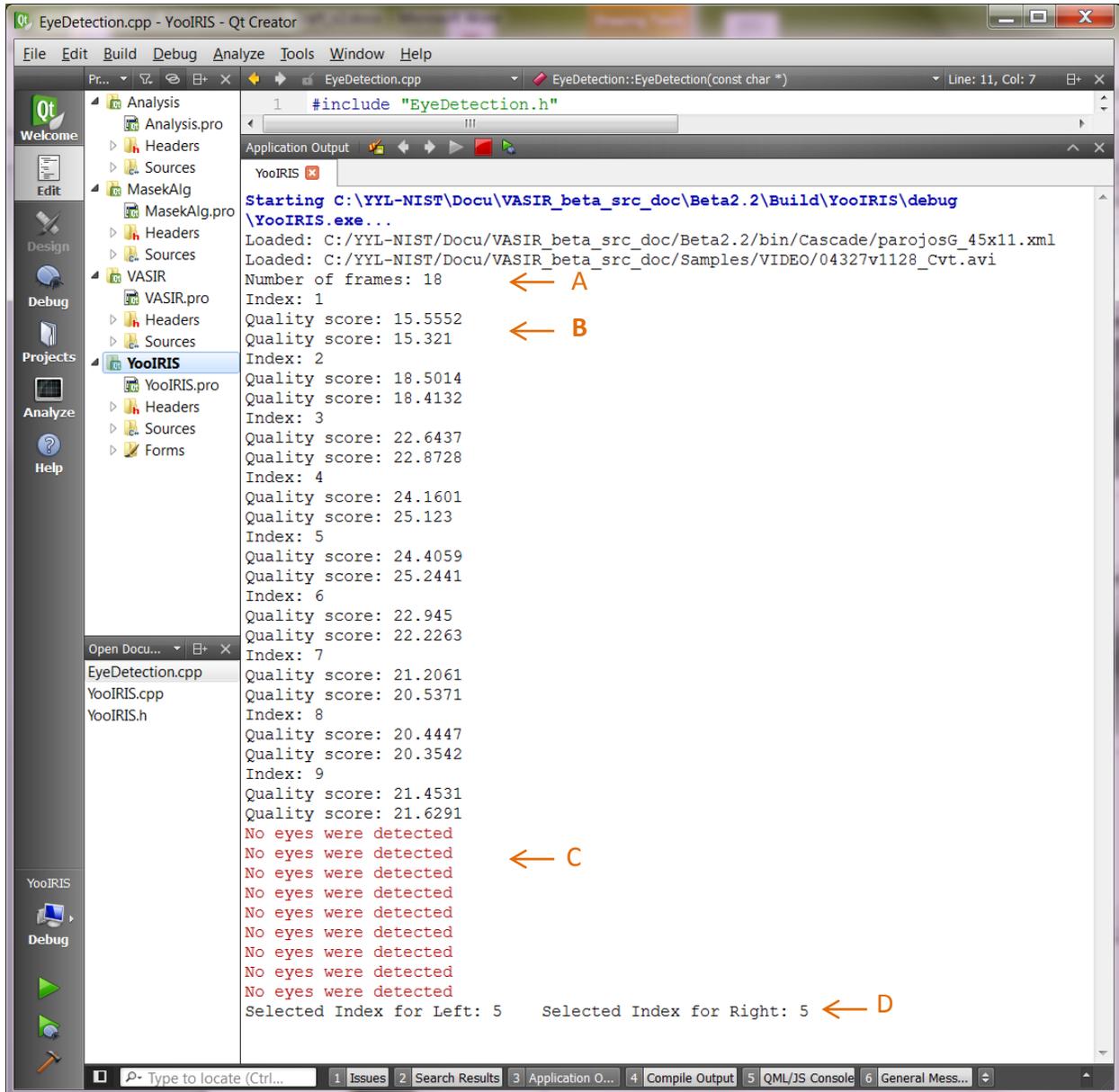
Depending on the dataset, the installation of additional codecs may be necessary.

Click “Open FACE AVI” and select a face visible video file to start the eye regions detection.



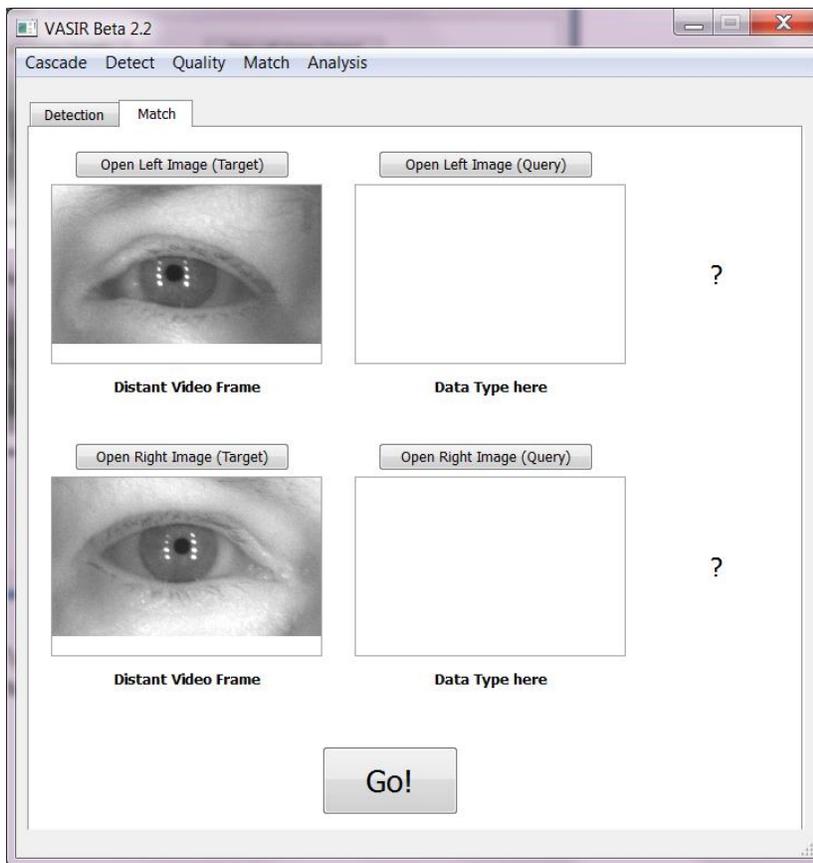
Once the detection starts, the “Detection” tab will display all video frames overlaid with the eye region detecting process information.

Relevant messages are displayed in the Application Output (or “Console”) below:



- A. The number of total frames in the selected video file
- B. The image quality scores for the left and right iris images
- C. A warning if no eyes were detected
- D. The selected index of the best quality image

After a successful run, the system automatically selects the best quality images for the left and right eye based on the calculated quality scores, and displays these selected images in the “Match” tab.



Furthermore, the “Data Type here” placeholder text is changed to “Distant Video Frame” since the imagery originated from a video file. For more details refer to Section “Match tab”.

Extracting the eye region image

The system automatically extracts the left and right eye images and saves them in the same folder as the original video file. The naming format is:

“<VideoFileName><Frame #>_<L(left)|R(right)><Sequence #>.bmp”

Name	Size	Type
 Test_Video.avi	110,597 KB	IrfanView AVI File
 Test_VideoF1_L1.bmp	306 KB	IrfanView BMP File
 Test_VideoF1_R1.bmp	306 KB	IrfanView BMP File
 Test_VideoF2_L2.bmp	327 KB	IrfanView BMP File
 Test_VideoF2_R2.bmp	327 KB	IrfanView BMP File

The system automatically selects the best quality iris images and saves them as follow:

“<VideoFileName><BEST>_<L(left)|R(right)><Sequence #>.bmp”

 Test_VideoBEST_R5.bmp	42 KB	IrfanView BMP File
 Test_VideoBEST_L5.bmp	42 KB	IrfanView BMP File

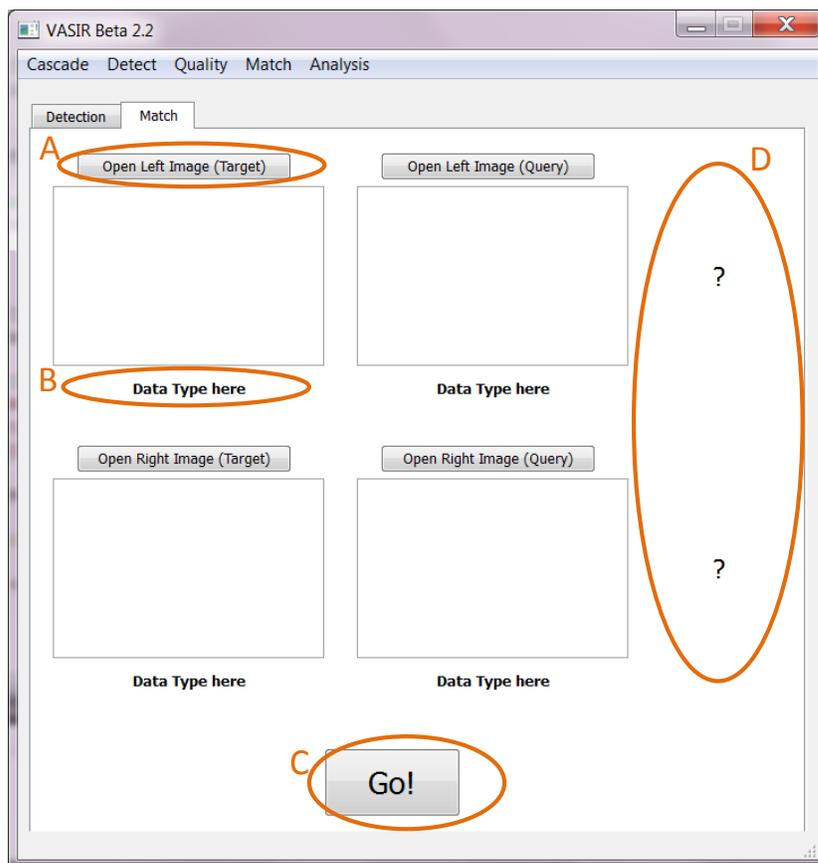
Match tab

The “Match” tab is used for iris verification (1:1).

If you would like to match a still image only (still-to-still), you can use this tab without having to load a cascade and the video file firsthand.

VASIR allows multiple scenarios to verify two biometric samples - depending on your purpose:

- Distant Video to Distant Video
- Distant Video to Classical Still
- Classical Still to Classical Still
- Classical Still to Distant Video



A quick glance at the screenshot above:

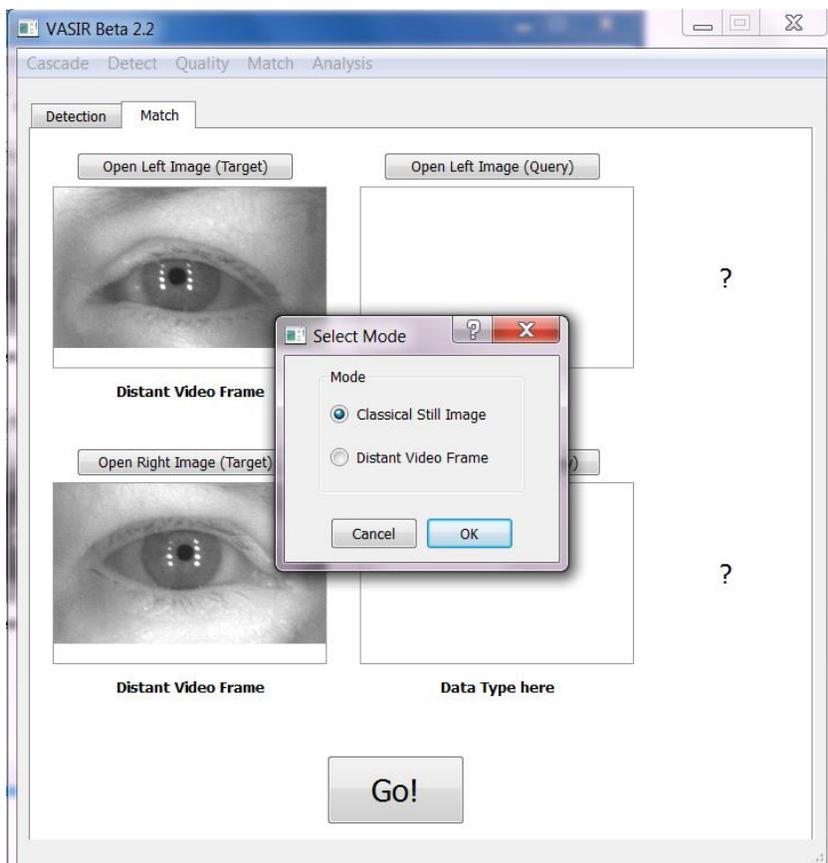
- A. Click this button – or any other “Open ...” button - to load an iris image. The image will be displayed (downscaled if necessary) within the box below the button.
- B. The encircled label will display the data type once you’ve chosen a mode in the “Select Mode” dialog.
- C. Click this button to match the target and the query biometric sample images.
- D. This area will contain the matching results for the left and right iris images.

Upon clicking the “Open Left (or Right) Image” button, you will be shown a dialog titled “Select mode” that contains two radio buttons: “Classical Still Image” and “Distant Video frames”.

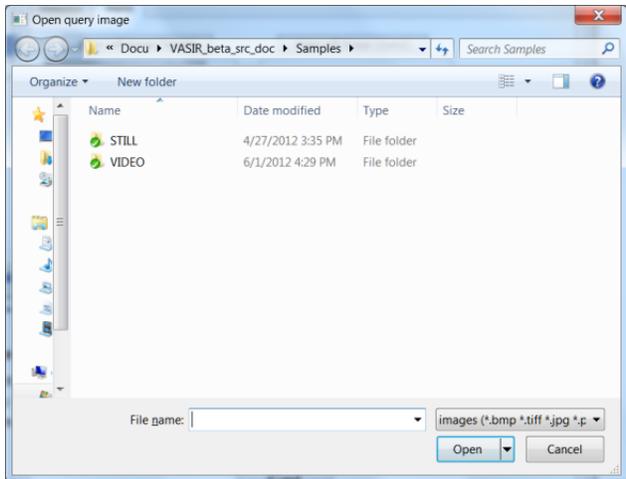
The “Classical Still Image” radio button indicates that the iris image is of decent quality. It was taken by a system akin to an LG2200 or LG4000.

The “Distant Video Frame” indicates the iris image is actually a video frame, captured by the Iris on Move (IOM) system at a certain distance.

Pick the mode type and then click “OK.”

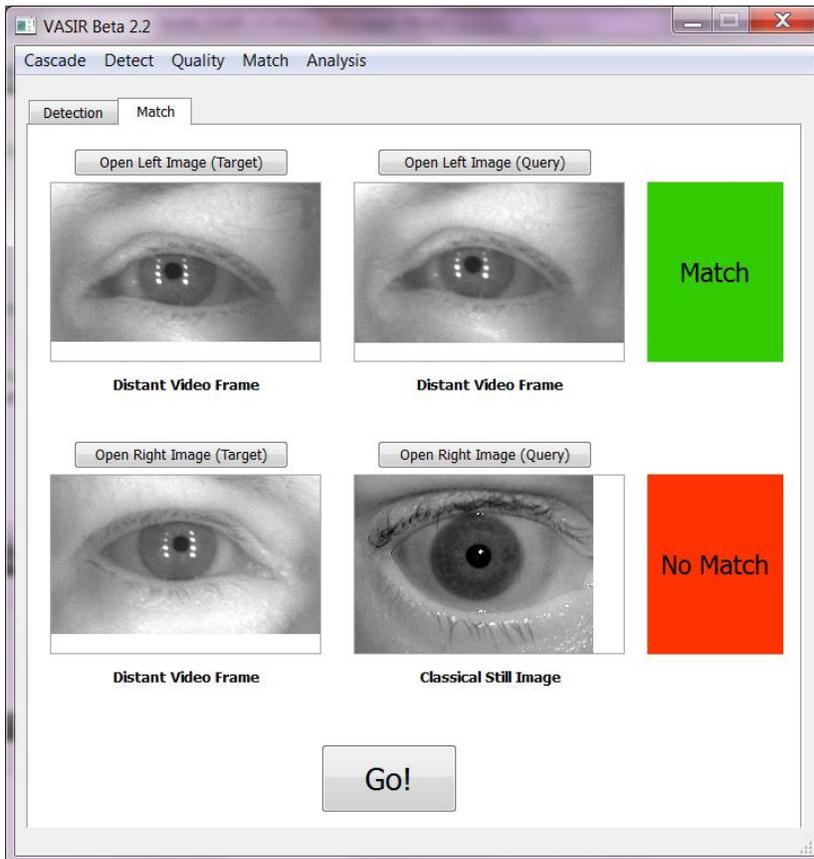


At this point VASIR loads the iris image; supported formats are BMP, TIFF, JPG, and PGM.



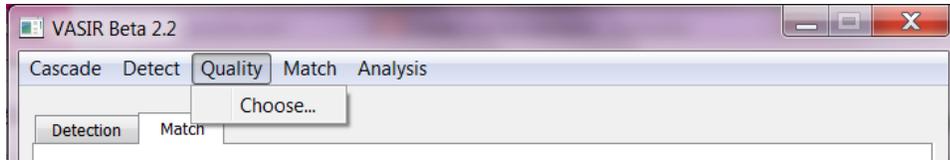
Next, click “Go!” to get the matching results.

The screenshot below shows an example of a Distant Video Frame vs Classical Still Image matching scenario. The two iris images for the left eye (top) are from the same person while the two images for the right eye (bottom) are from a different person.



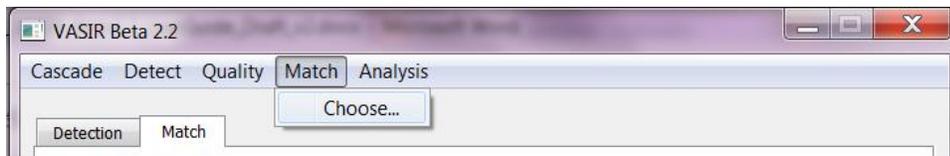
Quality menu

Click the “Choose” menu item to assess the quality of an image. You will see the calculated quality score in the Application Output.

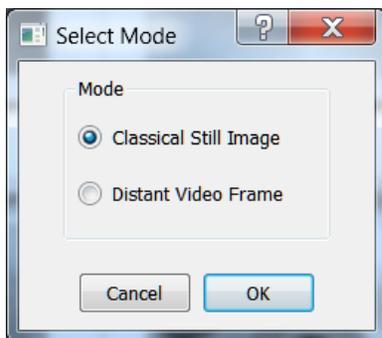


Match menu

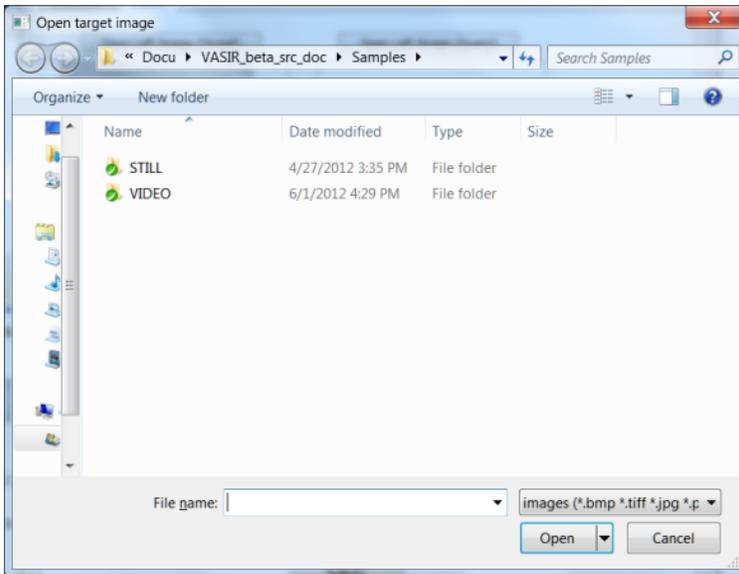
- 1) Click “Choose” to match a target still image and a query still image.



- 2) Select the mode for the target image.



- 3) Load the target image.

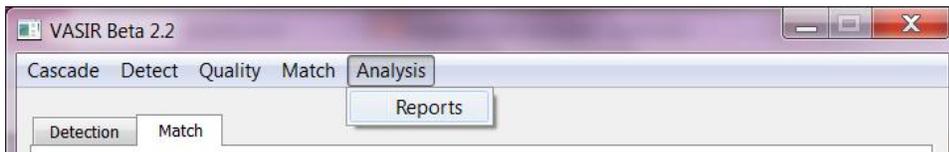


4) Repeat steps 2 and 3 for the query image.

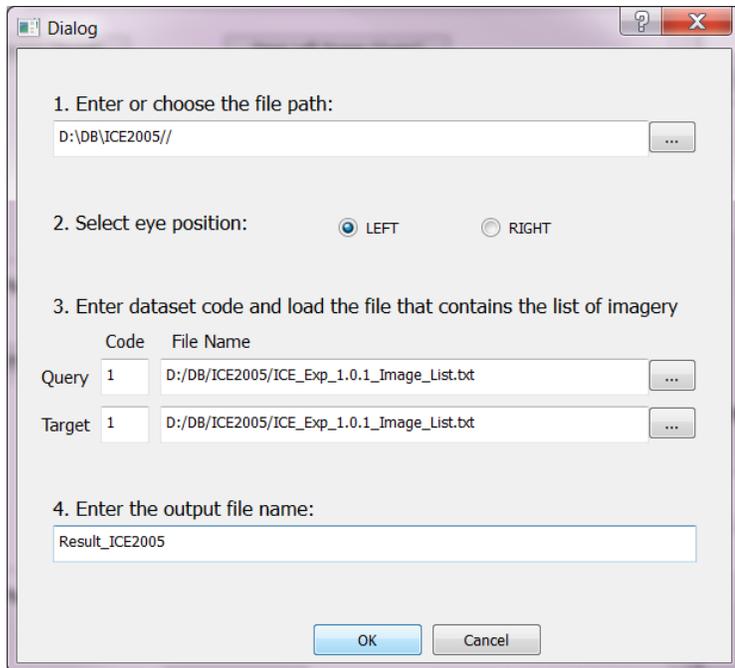
You will see the matching results in the “Match” tab.

Analysis menu

The “Analysis” menu allows the user to calculate matching (genuine) and nonmatching (imposter) scores, and performance measure scores. This function will generate four documents: (1) matching scores, (2) nonmatching scores, (3) ROC (Receiver Operating Characteristic) (4) reduced ROC.



Click “Reports” to produce the genuine scores, and then enter the inputs below. For example:



- 1) Enter or choose the file path
- 2) Select eye position (“Left or “Right”)
- 3) Enter data type “Code” for Query and Target:

- 1: ICE2005-LG2200
- 2: MBGC-LG2200
- 3: MBGC-IOM
- 4: ND2011-LG4000
- 5: ND2011-LGiCAM
- 6: ND2011-IGAD100
- 7: ND2011-CFAIRS

See the details at “Y. Lee, J. J. Filliben, R. J. Micheals, P. J. Phillips, M. D. Garris, “A Baseline for Assessing Biometrics Performance Robustness: A Case Study across Seven Iris Datasets” in 6th IEEE International Conference on Biometrics: Theory, Applications and Systems, 2013, pp. 1–8”.

- 4) Load the file that contains the list of image file names for Query and Target
- 5) Enter the matching output file name
- 6) Click “OK”. Four reports will be generated. For examples,
 - (1) Result_ICE2005_Left_Index_Matching.txt
 - (2) Result_ICE2005_Left_Index_NonMatching.txt
 - (3) Result_ICE2005_Left_Index_ROC.txt
 - (4) Result_ICE2005_Left_Index_ReducedROC.txt
- 7) After finishing all process, you will see “It is done...” message in Application Output.

Annex A. Performance on Datasets

Datasets

For the purpose of the benchmark evaluation, we examined the VASIR system performance using datasets collected by ICE (Iris Challenge Evaluation) 2005, MBGC (Multiple Biometric Grand Challenge), and NDSpring 2011.

For further information and a downloadable version of the dataset, please refer to the website:

<http://www.nist.gov/itl/iad/ig/ice.cfm>

<http://www.nist.gov/itl/iad/ig/mbgc.cfm>

Performance Evaluation

The detailed evaluation protocol and performance for each dataset are described in the publications below.

- 1) Yooyoung Lee, Ross J. Micheals, P. J. Phillips, James J. Filliben, “VASIR: An Open-Source Research Platform for Advanced Iris Recognition Technologies”, Journal of Research, National Institute of Standards and Technology (JRNIST), V118, p218-259, 2013.
<http://dx.doi.org/10.6028/jres.118.011>
- 2) Y. Lee, J. J. Filliben, R. J. Micheals, P. J. Phillips, M. D. Garris, “A Baseline for Assessing Biometrics Performance Robustness: A Case Study across Seven Iris Datasets” in 6th IEEE International Conference on Biometrics: Theory, Applications and Systems, 2013, pp. 1–8.

If you need the similarity matrices and ROC (Receiver operating characteristic) scores for comparing your performance, please send an email to vasir<NO SPAM>@nist.gov.

Annex B. Doxygen description of source code

Please have a look at the “NIST_VASIR_Beta2.0_doxygen_html” document on our website:

<http://www.nist.gov/itl/iad/ig/vasir.cfm>

Acknowledgement

We would like to thank Karen Marshall for her valuable comments on this user guide.